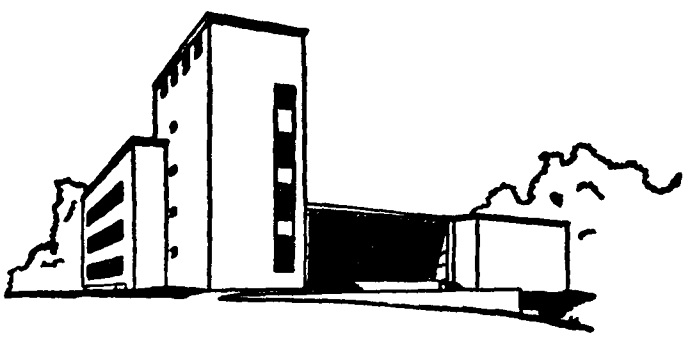
Carnegie-Mellon University

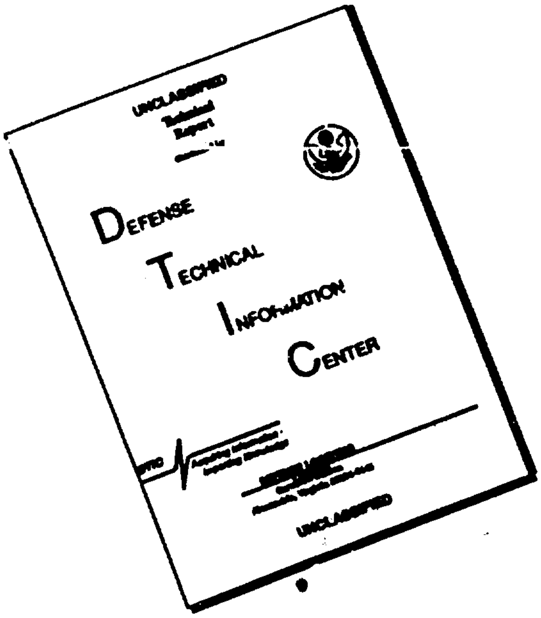
PITTSBURGH, PENNSYLVANIA 15213

GRADUATE SCHOOL OF INDUSTRIAL ADMINISTRATION

WILLIAM LARIMER MELLON, FOUNDER



DISCLAIMER NOTICE



THIS DOCUMENT IS BEST QUALITY AVAILABLE. THE COPY FURNISHED TO DTIC CONTAINED A SIGNIFICANT NUMBER OF PAGES WHICH DO NOT REPRODUCE LEGIBLY.

Security Classification

DOCUMENT CONTROL DATA - R & D

(Security classification of title, body of abstract and indexing annotation must be entered when the overall report is classified)

1. ORIGINATING ACTIVITY (Corporate author) Graduate School of Industrial Administration Carnegie-Mellon University	2a. REPORT SECURITY CLASSIFICATION Unclassified 2b. GROUP Not applicable

3. REPORT TITLE

Generating All the Faces of a Polyhedron

4. DESCRIPTIVE NOTES (Type of report and inclusive dates)

Management Sciences Research Report June 1971

5. AUTHOR(S) (First name, middle initial, last name)

Claude-Alain Burdet

6. REPORT DATE	7a. TOTAL NO. OF PAGES	7b. NO. OF REFS
June 1971	20	4

8a. CONTRACT OR GRANT NO. N00014-67-A-0314-0007 b. PROJECT NO. NR -47-048 c. d.	9a. ORIGINATOR'S REPORT NUMBER(S) Management Sciences Research Report No. 271 9b. OTHER REPORT NO(S) (Any other numbers that may be assigned this report) W.P. 90-70-1

10. DISTRIBUTION STATEMENT

This document has been approved for public release and sale; its distribution is unlimited.

11. SUPPLEMENTARY NOTES	12. SPONSORING MILITARY ACTIVITY
	Logistics and Mathematical Statistics Br. Office of Naval Research Washington, D. C. 20360

13. ABSTRACT

The determination of all the extreme points of a given convex polyhedron $P \subset \mathbb{R}^n$ generally requires a substantial amount of computations; this note presents a conceptually simple algorithm for this purpose. Unlike other methods, the procedure generates only those basic solutions which are extreme points (i.e., only feasible basic solutions).

More generally, this approach is able to generate all the faces of any dimension k $(0 \leq k \leq n)$, that is all those k-dimensional subpolyhedra which lie on the boundary of the given polyhedron P.

Unclassified
Security Classification

14 KEY WORDS	LINK A		LINK B		LINK C	
	ROLE	WT	ROLE	WT	ROLE	WT
extreme point programming						
non-convex programming						
concave programming						
integer programming						
polyhedral sets						
vertex enumeration						
facial decomposition						
degeneracy						

DD FORM 1473 (BACK)

Unclassified
Security Classification

W.P. 90-70-1

Management Sciences Research Report No. 271

GENERATING ALL THE FACES OF A POLYHEDRON

by

Claude-Alain Burdet

June 1971

This report was prepared as part of the activities of the Management Sciences Research Group, Carnegie-Mellon University, under Contract N00014-67-A-0314-0007 NR 047-048 with the U. S. Office of Naval Research. Reproduction in whole or in part is permitted for any purpose of the U. S. Government.

Management Sciences Research Group
Graduate School of Industrial Administration
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

ABSTRACT

The determination of all the extreme points of a given *convex polyhedron* $P \subset \mathbb{R}^n$ generally requires a substantial amount of computations; this note presents a conceptually simple algorithm for this purpose. Unlike other methods, the procedure generates only those *basic solutions* which are *extreme points* (i.e., only feasible basic solutions).

More generally, this approach is able to generate all the *faces* of *any dimension* k $(0 \leq k \leq n)$, that is all those k-dimensional subpolyhedra which lie on the boundary of the given polyhedron P.

Introduction

Polyhedral sets are the most widely used constrained sets in mathematical programming and they are usually defined by a system of linear (in)equalities. In fact, other characterizations of these sets are often impractical in the convex programming context.

However, as mathematical programming penetrates into the darker and less structured areas of non-convex programs (concave programming or discrete (in particular, integer or zero-one) programming, for instance), more information on the polyhedral set P of feasible solutions is required.

A classical question is that of finding the vertices of P, and seems very difficult to answer practically in large dimensional vector spaces. One may also be interested in the complete face structure of P, that is, in finding a characterization of each k-dimensional face F of P, for $0 \leq k \leq n$. (Clearly the quest for vertices is a special case of the latter, since they are 0-dimensional faces of P.)

This paper presents a simple approach to the determination of the face structure of a polyhedron. An algorithm is presented, which generates all the desired information concerning the complete lattice of faces of a polyhedron P, in the form of a non-redundant facial arborescence.

Some applications are mentioned and described in general terms in the last section. One particular case is that of general quadratic programming, which is the object of the follow-up paper [4].

Section 1. Minimal Sets.

1.1 Consider the polyhedral set P, defined by the following system of inequalities

$$x_i = a_{io} + \sum_{j \in N} a_{ij} x_j \geq 0 \ , \ i \in M \tag{1}$$

with non-basic index set $N = \{1,2,\ldots,n\} \subset M = \{1,2,\ldots,n,n+1,\ldots,n+m\}$.

- The matrix A contains an n-by-n identity submatrix, which corresponds to the constraints $x_j \geq 0,\ j \in N \subset M$
- For simplicity, we only consider here the case where P has full dimension n and is bounded.

Definition: A subset $I \subset M$ is called minimal if

$$\{x / x_i \geq 0,\ \forall i \in I\} \subset \{x / x_i \geq 0,\ \forall i \in M\} = P$$

and for every $i_o \in I$, there exists a point $\bar{x}$ such that

$$\bar{x}_{i_o} < 0 \quad \text{and} \quad \bar{x}_i \geq 0 \ , \ \forall i \in I - \{i_o\} \tag{2}$$

Property 1: The constraints $x_i \geq 0$, $\forall i \in (M - I)$ are redundant and one has

$$P = \{x / x_i \geq 0 \ , \ \forall i \in I\} \tag{3}$$

<u>Property 2</u>: For every $i_o \in I$, the hyperplane $x_{i_o} = 0$ contains a $(n-1)$-dimensional facet of P.

<u>Proofs</u>:

to 1: By definition one has

$$\forall\, q \in (M-I): x_q \geq 0 \;,\; \forall\, x \in \{x \mid x_i \geq 0 \;,\; \forall\, i \in I\} \qquad \text{q.e.d.}$$

to 2: The set $D = \{x \mid x_i \geq 0 \;,\; \forall\, i \in I - \{i_o\}\}$ contains a point $\bar{x}$ with $\bar{x}_{i_o} < 0$, because $i_o \in I$. Furthermore there exists a point $\tilde{x}$ in the interior of D with $\tilde{x}_{i_o} = 0$; such a point may be constructed by choosing an arbitrary interior point $\hat{x} \in \mathrm{Int}(P)$ and intersecting the line $\hat{x} - \bar{x}$ with the plane $x_{i_o} = 0$, yielding $\tilde{x}$. Since $P \subset D$ one has $\tilde{x} \in \mathrm{Int}(D)$ due to the convexity of D, i.e. $\tilde{x} = (1-\mu)\,\hat{x} + \mu\,\bar{x}$, with $0 \leq \mu < 1$. Moreover, since $P \subset D \subset \mathbb{R}^n$, the polyhedral set D has the same dimension n as the set P.

Take now a (small enough) open n-dimensional ball $B(\tilde{x}) \subset \mathrm{Int}(D)$, containing $\tilde{x}$, and consider the intersection $B' = \{x \in B(\tilde{x}) \mid x_{i_o} = 0\}$: by construction B' is a $(n-1)$-dimensional open ball $\subset D$, and it lies in the intersection of P with the hyperplane $x_{i_o} = 0$, i.e. in a facet of P, which has dimension $(n-1)$. q.e.d.

1.2 The concept of minimal set of inequalities I provides for the basic tool of an algorithm for the face decomposition of P ; I can be obtained by the following:

PROCEDURE MINSET

Give a system of inequalities (1), i.e., a matrix A with (n+m) rows and (n+1) columns, containing an identity submatrix; the procedure MINSET determines the minimal set $I \subset M$. A primal feasible linear programming tableau for the system (1) is required to start MINSET. During the execution of MINSET, the elements $i_o \in M$ are selected one after the other, and the corresponding row

$$x_{i_o} = a_{i_o0} + \sum_{j \in N} a_{i_oj} x_j$$

is (momentarily) chosen as objective function. Optimization of the following L.P.

Minimize x_{i_o}, subject to $x_i \geq 0$, $\forall i \in R - \{i_o\}$ furnishes a minimal value $\bar{x}_{i_o}$; the index set R is determined by the procedure and satisfies $I \subset R \subset M$. If $\bar{x}_{i_o} < 0$ then one has $i_o \in I$, by definition of the minimal set I ; if $\bar{x}_{i_o} > 0$, then the constraint $x_{i_o} \geq 0$ is considered redundant and is disposed of; the case $\bar{x}_{i_o} = 0$ is treated separately.

MINSET: 1 Set $R = M$ and $I = I_o = \emptyset$

2 Choose a basic index $r \in \left(R - (R \cap N)\right)$;

- if $R = \emptyset$ then STOP!
- if $R \neq \emptyset$ but $R \subset N$, then change the basis (and the non-basic set N) by choosing a positive pivot which preserves primal feasibility; if there exists no such element in the current tableau then replace I by $I \cup R$ and STOP!

3 Set $R = R - \{r\}$ and consider the r^{th} row

$$x_r = a_{ro} + \sum_{j \in N} a_{rj} x_j$$

4 Solve by L.P. optimization the problem

Minimize x_r , s.t. $x_i \geq 0$, $\forall\, i \in (R \cup I)$ (4)

5 If the minimal value of x_r is > 0 then go to 2.

If the minimal value of $x_r = 0$, then replace I_o by $I_o \cup \{i_o\}$ and go to 2.

6 Replace the I by $I \cup \{r\}$ and go to 2.

Because the procedure MINSET will be frequently called in the algorithm it is well worth noting the following remarks to speed up its execution:

1) In order to minimize the number of pivotal operations, the choice of r in Step 2 should correspond to a row x_r with the smallest possible number of <u>negative</u> elements a_{rj} (usually with just one $a_{rj} < 0$).

2) Suppose that, in the course of the (primal) optimization of Step 4, a <u>row</u> x_s, $s \in R$, with $a_{sj} \geq 0$, $\forall j$ is found in the current basic set; then s may be immediately discarded from the set R.

<u>Proof</u>: In the current basis, the condition $a_{sj} \geq 0$ is the optimality criterion of the L.P.

Minimize x_s, subject to $x_i \geq 0$, $\forall i \in ((R \cup I) - \{s\})$.

Thus, one may bypass the minimization of x_s and go to the steps 5 and 6, with $x_s = a_{so}$.

3) Similarly for the <u>column</u> of a non-basic variable x_j with $j \in R$, one may eliminate the element j from the set R. Define

$$\Delta = \min_{k/a_{kj} > 0} \left\{ \frac{a_{ko}}{a_{kj}} \right\}, \; k = \text{basic index (also } k = r)$$

Then j may be eliminated when $\Delta > 0$.

(A particular such case is when $a_{kj} \leq 0$, $\forall k$).

Proof: The preceeding condition is such that the solution remains feasible when x_j assumes a negative value $0 > x_j \geq -\Delta$.

Let us now show that the procedure MINSET does indeed determine a minimal set I_{min}:

(i) By construction, every point $x \in P' = \{x \mid x_i \geq 0,\ i \in I_{min}\}$ satisfies $x_i \geq 0$, $\forall i \in (M-I_{min})$; hence, $P' \subset P = \{x \mid x_i \geq 0, \forall i \in M\}$.

(ii) For every $i_o \in I_{min}$, the procedure MINSET constructs a point $\bar{x}$ with $\bar{x}_{i_o} < 0$ and $\bar{x}_i \geq 0$, $\forall i \in (R \cup I)$ where R and I are the current sets of step 4; since $(R \cup I) \supset I_{min}$ by construction, one has $\bar{x}_i \geq 0$, $\forall i \in I_{min} - \{i_o\}$.

(iii) The set I_o ($\neq \emptyset$) indicates that the given system (1) is degenerate, i.e., that some k-dimensional faces ($0 \leq k \leq n$) of the polyhedron P are contained in more than (n-k) hyperplanes $x_i = 0$, $i \in M$; this situation does not affect the minimal property of I_{min}, but the identification of the elements of I_o is important in order to eliminate redundancy of the facial decomposition. (See section 3.5).

(iv) Property 3: In the non-degenerate case ($I_o = \emptyset$), the minimal set is unique.

Proof: For every $i \in (M-I)$ one has $x_i \geq \epsilon > 0$, $\forall x \in P$ since $I_o = \emptyset$ by hypothesis; this is a well defined criterion which divides M uniquely in two disjoint subsets (M-I) and I.

Section 2: The Faces of P .

At the beginning, one applies MINSET to the system (1), that is, to the given polyhedron P in order to determine the set I (degeneracy will be considered separately in 3.5). But from the theory of polyhedral sets, one knows that every face F of the polyhedron P is a polyhedron. Thus, MINSET can be applied to the faces F of P as well. In particular, to the faces $F(i_1)$, $F(i_1,i_2)$,... where

$F(i_1) = \{x \in P \mid x_{i_1} = 0 \ , \ i_1 \in I_o\}$, with a corresponding minimal set $I(i_1) \subset I$,

$F(i_1,i_2) = \{x \in P \mid x_{i_1} = x_{i_2} = 0 \ , \ i_1 \in I, \ i_2 \in I(i_1) \subset I \}$ " $I(i_1,i_2) \subset I(i_1)$,

etc....

A sequence of faces $F(i_1)$, $F(i_1,i_2)$,...,$F(i_1,\ldots,i_q)$ is generated with the following properties.

Property 3: For $k + s \leq n$, one has

$$I(i_1,i_2,\ldots,i_s) \supset I(i_1,i_2,\ldots,i_s,i_{s+1},\ldots,i_{n-k})$$

$$\text{and} \quad F(i_1,i_2,\ldots,i_s) \supset F(i_1,i_2,\ldots,i_s,i_{s+1},\ldots,i_{n-k}) \ .$$

Proof: by construction.

In conclusion, one sees that repeated use of the procedure MINSET, leads to the construction of an aborescence with initial node P (n-dimensional face) itself; at the level below, one finds all the (n-1)-dimensional faces of P (one for each $i \in I$); then, below, the faces of these faces (i.e., the (n-2)-dimensional faces of P) etc.,... At every node (i.e., face $F(i_1,\ldots,i_q)$) the minimal set $I(i_1,\ldots,i_q)$ determines the

branches (how many and which) leading to the level below. At the lowest level, one ultimately finds the vertices of P.

The next section presents an algorithmic construction of this arborescence.

Section 3: An exhaustive arborescence for the faces of P.

3.1 The following procedures TREE, FACE2D, SIMFACE and BACKTRACK generate one by one a list of n-arrays called VERTEX = $\{i_1, i_2, \ldots, i_n\}$.

In this version, the algorithm requires the storage of

- the arrays COL [t;] , t = 1,2,...,n which have at most m components.
- the n-arrays N , M and MI , J
- the "dynamic" arrays (at most m components each): I = I[0;], I[1;],...,I[n-1;] .
- the current linear programming tableaux A , which is at most (n+m) by (n+1).

TREE: 1 - Set t = 1 and J = ∅ ; the initial tableau stems from (1); I[0;] = I_o ; MI[0] = number of elements in I ; M[0] = 1

- Set COL [1]: = first column of the original matrix A (corresponding to the first non-basic variable x_j)
- Set N[1]: = j = non-basic index of the first column
- Delete the first column from the matrix A , to obtain the ament matrix A (which is thus an MI[0] by n array)

2 If (n-t) = 2 then use FACE2D and go to BACKTRACK.

3 Apply MINSET to the current tableau A : → I[t;]

- Set MI[t] = number of elements in the set I[t;]
- Set M[t] = 0

4 If MI[t] = 1 , apply SIMFACE and go to BACKTRACK.

5 If M[t] = MI[t] , go to BACKTRACK ;

6 - Set $M[t] := M[t] + 1$;

7 Take the $M[t]^{th}$ component j of the array $I[t;\ ]$ i.e., $j := I[t;M[t]]$;

8 Make x_j non-basic (if it is not already) preserving primal feasibility.

When pivoting is required, one must transorm also the columns stored in the arrays COL[s] , for all s, $1 \leq s \leq t$; this can be done by forming a full tableau TAB = {COL[1], COL[2],...,COL[t],A} ; TAB is then transformed by pivoting (pivot in A) and the new columns 1,... are stored again in COL[1],...,COL[t] .

- Set $t := t + 1$;
- Set $N[t] := j$;

9 - Set COL[t] := column of the current tableau A corresponding to the non-basic index j

- Set A = matrix obtained from the current matrix A by deleting the column j ;

10 Go to step 2.

BACKTRACK:

11 Set $t = t - 1$.

12 If $t = -1$ then STOP!

13 Adjoin the column COL[t] to the current matrix A , thus forming a new (augmented) matrix A .

3.2 We need the following special procedures in the above algorithm.

Two-dimensional faces: FACE2D.

Two-dimensional faces of the convex polyhedron P can be treated separately, because a straightforward sequence of pivots determines all their vertices and consequently also their 1-dimensional faces.

FACE2D:

1 Find a basic feasible solution, i.e., two non-basic indices n_1, n_2;
- Set $R = I[n-2;]$ and $J = \emptyset$

2 Register the corresponding vertex characterized by the non-basic index set VERTES = $\{N[1], N[2], \ldots, N[n-2], n_1, n_2\}$.

3 Choose a non-basic index $j = n_1$ or n_2 ;
- if both n_1 and n_2 belong to J then STOP!

4 Set $J := J \cup \{j\}$ and $R = R - \{j\}$

5 Find the basic index $i \in R$ which is to leave the basis when the non-basic index j enters the basis (in order to maintain primal feasibility).

6 Pivot (on a_{ij} and go to 2.)

3.3 Simplical faces: SIMFACE

When the minimal set I of a k-dimensional face F consists of (k+1) elements, F is a simplex and its (k+1) vertices can be immediately determined, without resorting to subfaces of F .

Let $I = I(i_1, i_2, \ldots, i_{n-k}) = \{j_1, j_2, \ldots j_{k+1}\}$;

SIMFACE: 1 Find a first basic feasible solution, i.e.
k non-basic indices $n_1,\ldots,n_k \in I$ and
store VERTEX = $\{N[1],N[2],\ldots,N[n-k],n_1,\ldots,n_k\}$

2 For each $i = 1,\ldots,k$, generate a new non-basic array VERTEX from the one obtained in step 1 above by replacing the index n_i by the last remaining element $n_{k+1} \in I$. Clearly n new arrays VERTEX are generated in this fashion.

The procedure SIMFACE only determines the vertices of F, but if all substances of F are desired, they can be obtained in a similar manner.

3.4 The algorithm TREE, BACKTRACK, FACE2D, and SIMFACE determines a non-redundant list of non-basic indes sets VERTEX which contains <u>all</u> the vertices of P.

<u>Proof</u>: The procedure TREE generates all sets of indices $N = \{N[1],N[2],\ldots,N[t]\}$ such that $N[i] \in I[i;\ ]$, for all $1 \leq i \leq t$ and for all t, $0 \leq t \leq n-2$; for all $t \geq 0$; for $t = n-2$ the procedure FACE2D takes over and finds all vertices in that face ; occasionally for $0 \leq t < n-2$, the procedure SIMFACE will do the same. Hence <u>all basic feasible solutions</u> of P with non-basic variables in the minimal set I (of P) are generated because by construction,

$I = I[0;] \supset I[1;] \supset \ldots \supset I[t;]$ always holds true.

Furthermore FACE2D and SIMFACE determine only basic feasible solutions.

Moreover this list is non-redundant, because step 6 of TREE guarantees that the same index set N is generated once and only once. q.e.d.

3.5 Degeneracy: When some faces (or vertices) of P are degenerate, it may happen that geometrically identical faces of P are algebraically represented by different sets N : such faces will then appear more than once in the arborescence (but each time with a different index set N, i.e., for instance

$$\{x \mid x \in F_1\} = \{x \mid x \in F_2\}$$

with $F_1 = F(i_1, i_2, \ldots, i_q)$

$F_2 = F(j_1, j_2, \ldots, j_q)$

and $\{i_1, i_2, \ldots, i_q\} \neq \{j_1, j_2, \ldots, j_q\}$

This redundancy is due to degeneracy in the arborescence, i.e., to the fact that some faces $F(i_1, \ldots, i_q)$ are "over-determined" in the sense that $\exists i \notin \{i_1, \ldots, i_q\}$ such that $F(i_1, \ldots, i_q) \subset \{x \mid x_i = 0\}$.

Such an index i is identified by the procedure MINSET and one has

$$i \notin I(i_1, \ldots, i_q) \text{ but } i \in I_o(i_1, \ldots, i_q).$$

Hence all the characterizations by different index sets $\{i_1,\ldots,i_q\}$ of the same face F can easily be obtained from the index set $D_F = \{i_1,\ldots,i_q\} \cup I_o(i_1) \cup I_o(i_1,i_2) \cup \ldots$ $\ldots \cup I_o(i_1,\ldots,i_q)$; One has by definition

$$D_F = \{i \mid x_i = 0, \forall x \in F\} \subset M$$

and $Aff(F) = \{x \mid x_i = 0, \forall i \in D_F\}$. The corresponding linear system

$$x_i = a_{i_o} + \sum_{j \in N} a_{ij} x_j = 0 \ , \quad i \in D_F \subset M$$

has rank q ($\leq n$); it is over-determined in the sense that contains $d_F > q$ linearly dependent equations (d_F = number of elements in D_F) whenever one of the sets $I_o \neq \emptyset$.

Every set of q linearly independent rows represents a characterization of F which appears in the arborescence and causes redundancy; however, because the main subroutine TREE generates faces in a lexicographically increasing order, redundancy can easily be eliminated by the following additional bookkeeping:

D1 - Generate the <u>first</u> (lexicographically) degenerate representation, i.e., the next redundant node to be encountered by the procedure TREE;

D2 - Store the lexicographic order rank r of that node;

D3 - When BACKTRACK attains the order r (terminate that redundant branch and) go to D1 in order to update r;

Note that the degeneracy sets D_F are generated in increasing lexicographic order; and the step D1 does not require any additional computations; moreover step 3 does eliminate redundancy because the degeneracy sets newly discovered by the procedure TREE all belong to higher orders than the current one (i.e., one never finds out in a later phase that some previously enumerated faces were redundant).

Section 4: Some applications of a Facial Optimization Method (FOM) for linearly constrained mathematical programming.

The Facial Optimization Method (FOM) is of the branch and bound type, where the branching procedure is generated according to the facial structure of the constraint polyhedron P ; in this manner primal feasibility is always present in all the nodes. It should be noted that the arborescence of section 2 still contains some useful flexibility in the choice of the order in which the variables are to be "blocked" at value 0 (as non-basics); thus the usual preference rules of branch and bound algorithms also apply here. Since in every way the facial arborescence resembles other branching trees, the usual bound estimators also apply; clearly the efficiency of the facial approach as compared to other branch and bound algorithms depends on P and its structure.

In a vertex to vertex optimization method, where additional properties must be checked as in 4.1, one must be able to identify all neighbouring vertices of a given vertex $\bar{x}$, i.e. find all extreme rays of a (degenerate) cone. Consider a basis at $\bar{x}$

$$x_i = \bar{x}_i + \sum_{j\in N} a_{ij}\, x_j \geq 0 \ , \ \forall i \in M \tag{4}$$

with non-basic set $N \subset M$. For the rays stemming from $\bar{x}$, only the planes that are tight at $\bar{x}$ are of interest, i.e.

$$M' = \{i \mid i \in N \ , \ \text{or} \ \ \bar{x}_i = 0\} \subset M$$

The problem of finding the extreme rays of the cone C

$C = \{x \mid x_i \geq 0 \ , \ \forall i \in M'\}$ is equivalent to that of finding all extreme

points of the (n-1)-dimensional polyhedron $P = \{x \in C \mid \sum_{j \in N} x_j = 1\}$; the corresponding system (1) can be obtained in one pivot from (4). In [3] a search method of this type is developed for concave programs.

4.1 Zero-one programming: Here the polyhedron P is a subset of the unit cube; hence every feasible integer solution is a vertex of P. Starting at the L.P. optimal solution $\bar{x}$, a branching process which determines (not necessarily explicitly) all neighbours of $\bar{x}$, then all neighbours of these neighbours, etc....will terminate when an integer vertex has been found and all neighbours generated so far furnish a lesser value of the objective function. Depending on the structure of P, this branch and bound algorithm can use efficiently the facial structure of P, particularly for polyhedra P where $\bar{x}$ always has an integer neighbour.

4.2 General quadratic programming: Suppose we want to optimize (mximize or minimize) a quadratic objective function (not necessarily convex or concave) on a polyhedral set P. It can be shown that the candidates for the optimum are either vertices of P or (more generally) optimal solutions of the convex quadratic problems

$$\text{minimize} \quad f(x) \quad , \quad \text{s.t.} \quad x \in F$$

where F is a k-dimensional face of P such that the function f remains convex on F.

The facial decomposition method leads in this case to an efficient algorithm for general quadratic, linearly constrained problems which is presented in [4].

Of course FOM can be applied, conceivably, to many other special optimization problems over polyhedral sets (such as concave programming,

general non-linear programming, etc....) but it is not clear at this point where and when it furnishes an efficient approach in such general context.

5. Conclusions

The algorithm for generating all vertices of a polyhedron P presented in this note develops a facial arborescence which may be considered minimal, since every node is generated exactly once, and corresponds to a feasible face on the boundary of P . In particular, one can generate in this arborescence the set of vertices of P (i.e. feasible basic solutions) without generating any other basic solution, as is the case for methods [1] and [2]. One may argue that the additional pivoting, necessary for the construction of the minimal sets outweights this theoretical property; in view of the exponential growth of the face structure, however, this should not be expected to hold true in the general case. A program requiring a modest amount of storage (i.e., where practically only space for A is needed) is being implemented. When degeneracy occurs, a subtree of the facial arborescence can be eliminated by additional bookkeeping. The remaining arborescence is in one to one correspondence with the k-dimensional faces ($n \geq k \geq 0$) of P for all k. Precise comparisons in this context are not easily formulated mathematically and experimentation is currently under way with randomly generated, non-structured matrices A of a size up to 20 by 10.

REFERENCES

[1] Balinski, M.L., "An Algorithm for Finding All Vertices of a Convex Polyhedral Set", J. Soc. Indust. Appl. Mathematics, Volume 9, No. 1 (1961).

[2] Motzkin, T.S., Raiffa, H., Thompson, G.L., and Thrall, R.M., "The Double Description Method", Contributions to the Theory of Games (Vol. 2), H.W. Kuhn and A.W. Tucker (eds.), Annals of Mathematical Studies, No. 28, Princeton Press (1953).

[3] Burdet, C.-A., "A Combinational Approach to Linearly Constrained Concave Programming", Carnegie-Mellon University, Graduate School of Industrial Administration, W.P. 75-70-1.

[4] Burdet, C.-A., "General Quadratic Programming", Carnegie-Mellon University, Graduate School of Industrial Administration, W.P. 41-71-2.